Predicting Risky Clones Based on Machine Learning

Ayaka Imazato[1], Keisuke Hotta[1], Yoshiki Higo[1], and Shinji Kusumoto[1]

Graduate School of Information Science and Technology, Osaka University,
1-5, Yamadaoka, Suita, Osaka, Japan,
{i-ayaka, k-hotta, higo, kusumoto}@ist.osaka-u.ac.jp

Abstract. Code clones are similar or identical code fragments to one another in source code. It is said that code clones decrease maintainability of software. On the other hand, all the code clones are not necessarily harmful to software. In this study, we propose a method to identify risky code clones out of all the code clones in source code by using machine learning techniques. Our proposed method learns information about features of code clones which existed in the past and whether they were risky or not. Then, based on these information, we identify risky code clones. As a result of a pilot study, we confirmed that the proposed method was able to predict risky code clones with high accuracy.

1 Introduction

It is said that code clones (hereafter, clone) have bad effects on maintainability of software. For example, when one code fragment is modified, other code fragments that are similar or identical to it also require the same modifications frequently [2]. When multiple code fragments that are similar to one another require similar modifications, there is a possibility of overlooking some of the code fragments that should be modified. If overlooking happens, bugs might occur in the overlooked location. Hence, clones could decrease maintainability of software [1]. On the other hand, all the clones are not necessarily risky. For example, clones are harmless to maintainability of software if they have never been modified since they had appeared. Moreover, it is not realistic that developers manage all the clones because there are a huge number of clones.

It is necessary to take care of only risky clones out of all the clones in order to manage clones efficiently. In this study, we propose a method to identify risky clones out of all the clones by using machine learning (hereafter, ML). In our proposed method, we analyze development histories of software to obtain information about features of clones which existed in the past and whether they caused bugs or not. Then, based on these information, we construct models to predict risks of clones, and identify risky clones out of all the clones in current source code with the models. We have implemented our proposed method, and conducted a pilot study to evaluate the accuracy of predictions with our proposed method. As a result, we confirmed that the proposed method was able to predict risky clones with high accuracy.

2 Background and Related Work

All the clones are not necessarily risky. Hence, it is necessary to identify only risky clones in order to manage clones efficiently. The authors thought that it might be possible to predict risky clones with ML. ML is a technique to predict or identify the characteristics of unknown data by learning existing data. Several research uses ML to predict clones that should be taken care of.

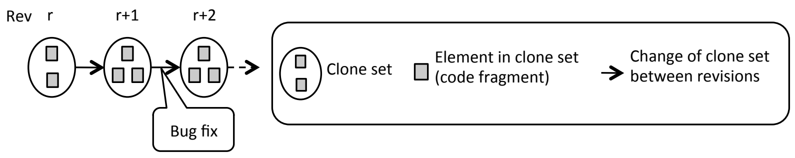
Yang et al. assume that judging whether a clone is useful or not varies from user to user [4]. They proposed a method to identify useful clones for each user with ML. Wang et al. proposed a method to predict risks of the clones when clones are generated by copy and paste operation [3].

In Yang's proposed method, users have to classify clones manually beforehand. Wang's method works only under the limited situations. On the other hand, our proposed method does not need any advance preparations by users, and can predict risks of arbitrary clones.

3 Proposed Method

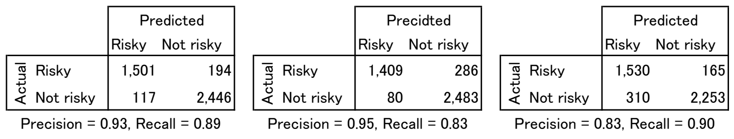
We propose a method to predict risks of clones by using ML. ML learns existing data, and predict or identify the characteristics of unknown data based on learned information. In ML, data for learning are called **training data**, and a model that is constructed by learning training data to predict characteristics of unknown data is called **learning model**. Our method learns information of *clone set* (group of code fragments that are similar to one another) as training data, and constructs a learning model to predict risks of clone sets. Also, our method intends to identify clone sets that will cause bugs in the future. Therefore, we define that clone sets which will cause bugs in the future are risky, and otherwise not risky. Our method takes the development history of the target software as its input, and provides a learning model to predict risky clone sets as its output. The proposed method consists of the following four phases.

1. First, we detect all the clone sets that were generated during past development process by analyzing the development history. Then, for each detected clone set, we obtain its evolutional data since it was generated. The evolutional data of clone set is typically called **genealogy of clone set** (**genealogy**). Fig.1 illustrates an example of it.
2. Then, for each genealogy, we judge whether it was risky or not. We regard a given genealogy as risky if it had undergone one or more bug fixes during its evolution. For example, the genealogy in Fig.1 is judged as risky because there was a bug fix between revision $r + 1$ and $r + 2$. Note that we get information of bug fixes through commit messages. We regard the modifications at the commit as bug fix if the commit message includes any words that imply to fix bugs such as *bug fix*.
3. After judging risks of all the genealogies, our method extracts training data from each genealogy. In our method, we extract the clone sets at the start



Rev r
r+1
r+2
Bug fix
Clone set
Element in clone set (code fragment)
Change of clone set between revisions

Fig. 1. An Example of a Genealogy



		Predicted	
		Risky	Not risky
Actual	Risky	1,501	194
	Not risky	117	2,446

Precision = 0.93, Recall = 0.89

(a) J48

		Precidted	
		Risky	Not risky
Actual	Risky	1,409	286
	Not risky	80	2,483

Precision = 0.95, Recall = 0.83

(b) BayesNet

		Predicted	
		Risky	Not risky
Actual	Risky	1,530	165
	Not risky	310	2,253

Precision = 0.83, Recall = 0.90

(c) SVM

Fig. 2. Result

revision of each genealogy as training data. For the genealogy in Fig.1, the clone set at revision r is extracted as training data. Subsequently, we judge risks of clone sets extracted as training data. In our method, training data belonging to risky genealogies are considered risky, and training data belonging to not risky genealogies are considered not risky. At the same time, we investigate the status of training data. The status of training data is, for example, the number of the elements that compose the clone set, the similarity between the elements, and so on. In this paper, we call such values and parameters that describe the status of clone set **feature value**. Our method uses 30 kinds of feature values in total.

4. Finally, for each clone set extracted as training data, we learn its feature values and information about whether it is risky or not together to construct a learning model. Our method predicts risk of a given clone set when users give feature values of it to the constructed learning model.

4 Pilot Study

We have conducted a pilot study with an open source project *jEdit* (the number of target revision is 5,292, and the development period is about 11 years) to evaluate our method. As described in the previous section, we detected all the genealogies and extracted their first clone sets from the *jEdit* project. We call the set of the clone sets data set. Note that the data set consists of 1,695 risky clone sets, and 2,563 not risky clone sets. This pilot study adopts the cross validation. The cross validation divides the data set into k blocks, and evaluates the accuracy of prediction by using these blocks. Note that the number of clone sets in each block is almost the same. In the cross validation, we use $k - 1$

blocks as training data, and the remaining one block as test data. Concretely, we construct a learning model by learning training data. Then, we adopt the learning model to test data and measure the accuracy. This process is repeated k times, with each of the k blocks used exactly once as test data. The average of k results is the entire accuracy. We set $k = 10$ in this pilot study. This pilot study uses two indicators below as measure for evaluation.

Precision: the rate of risky clone sets in clone sets that are predicted as risky by the learning model

Recall: the rate of clone sets that are correctly predicted as risky by the learning model in all the risky clone sets

Also, we use three algorithms (J48, BayesNet(Bayesian Network), SVM(Support Vector Machine)) to construct learning models, and evaluate each learning model. Fig.2 shows the results. As shown in this figure, the proposed method can predict risky clones with high accuracy, 83–95% *Precision* and 83–90% *Recall*. The accuracy of J48 is very high, both *Precision* and *Recall* are almost 90%. For SVM, *Precision* is the highest among the three algorithms, 95%, but *Recall* is the lowest. For BayesNet, on the contrary, *Precision* is lower than other algorithms, but *Recall* is the highest. The result of the pilot study showed that the proposed method was able to predict risky clones with high accuracy.

5 Conclusions

In this study, we proposed a method to identify risky clones out of all the clones in source code by using machine learning. As a result of a pilot study, we confirmed that the proposed method was able to predict risky clones with 83–95% *Precision* and 83–90% *Recall*. However, at present, our proposed method cannot rank specified risky clones, which is our future work.

Acknowledgment

This work was supported by MEXT/JSPS KAKENHI 24680002 and 24650011.

References

1. Higo, Y., Kusumoto, S.: How Often Do Unintended Inconsistencies Happen? - Deriving Modification Patterns and Detecting Overlooked Code Fragments-. In: ICSM. pp. 222–231 (Sep 2012)
2. Kim, M., Sazawal, V., Notkin, D., Murphy, G.: An Empirical Study of Code Clone Genealogies. FSE 30(5), 187–196 (Sep 2005)
3. Wang, X., Dang, Y., Zhang, L., Zhang, D., Lan, E., Mei, H.: Can I Clone This Piece of Code Here? In: ASE. pp. 170–179 (Sep 2012)
4. Yang, J., Hotta, K., Higo, Y., Igaki, H., Kusumoto, S.: Classification model for code clones based on machine learning. ESE pp. 1–31 (Jun 2014)